과제 #2
<총 **15**점>

- 총 1 문제: 최소 비용 신장 트리
  - 크루스칼 I & II 알고리즘: 5 점
  - 프림 알고리즘: 5 점
  - 솔린 알고리즘: 5 점

- 제출기한: 2022년 12월 5일 24:00 까지

- 제출물: MS VS 솔루션 폴더 (압축파일) 또는 소스 (압축) 파일

※ 충분한 주석은 필수입니다.
※ **제출 파일에 소스가 누락되지 않게 주의하세요.**

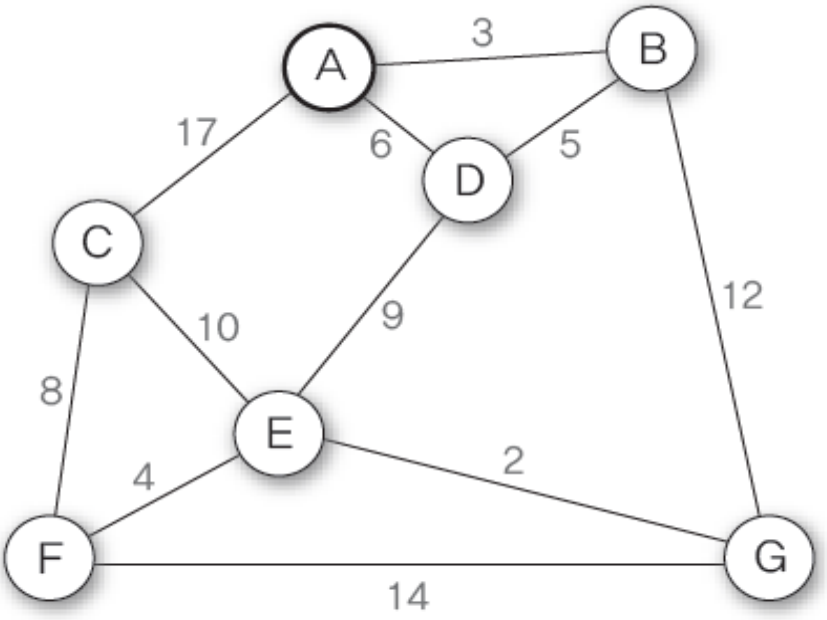
최소 비용 신장 트리
(Minimum-Cost Spanning Tree)

- 자료 구조 트리/그래프의 종합적 복습 겸 프로그래밍 역량 강화를 위한 과제입니다. 수업 시간에 다루었던, 아래의 4 가지 최소 비용 신장 트리 알고리즘을 C 프로그램으로 구현합니다.
  ① 크루스칼 I 알고리즘
  ② 크루스칼 II 알고리즘
  ③ 프림 알고리즘
  ④ 솔린 알고리즘

- 주의 사항: 구현 시 사용해야 할 필수 자료 구조 또는 알고리즘
  ① 크루스칼 I 알고리즘
    - 힙 (Max Heap): 최대 가중치를 가진 간선 삭제
    - 깊이/너비 우선 탐색 (DFS/BFS): 간선 삭제 시 단절 여부
  ② 크루스칼 II 알고리즘
    - 힙 (Min Heap): 최소 가중치를 가진 간선 삽입
    - 집합 (Set): 간선 삽입 시 Cycle 생성 여부
  ③ 프림 알고리즘
    - 집합 (Set): 성장하는 트리에 포함되는 정점들 확인
    - 큐 (Queue): 트리에 속한 정점의 부속 간선들의 가중치 비교
  ④ 솔린 알고리즘
    - 집합 (Set): 각 트리에 포함된 정점들 확인과 연결 시 합집합으로 트리 확장 (* find & union)
    - 큐 (Queue): 트리에 속한 정정의 부속 간선들의 가중치 비교

- 입/출력 예시: (다음 쪽으로)

※ 입력 그래프 (8장그래프, G10)



G10

인접행렬

```
#define MAX_VERTEX 7
#define INF 999
int g10[MAX_VERTEX][MAX_VERTEX] = {
        {  0,   3,  17,   6, INF, INF, INF},
        {  3,   0, INF,   5, INF, INF,  12 },
        { 17, INF,   0, INF,  10,   8,  INF},
        {  6,   5, INF,   0,   9, INF,  INF},
        {INF, INF,  10,   9,   0,   4,   2 },
        {INF, INF,   8, INF,   4,   0,  14 },
        {INF,  12, INF, INF,   2,  14,   0 }
};
```

※ 출력 예

```
*** MST: Kruskal I Algorithm ***
Heap[1~11]: [17] [14] [10] [9] [12] [6] [8] [3] [4] [2] [5]
   ... deleting the largest edge ...
- Edge (0 2)[17] deleted.
- Edge (5 6)[14] deleted.
- Edge (1 6)[12] deleted.
- Edge (2 4)[10] deleted.
+ Edge (3 4)[ 9] not deleted.
+ Edge (2 5)[ 8] not deleted.
- Edge (0 3)[ 6] deleted.
Heap[1~4]: [5] [4] [2] [3]

*** MST: Kruskal II Algorithm ***
Heap[1~11]: [2] [3] [6] [5] [4] [10] [8] [17] [9] [12] [14]
... inserting the smallest edge ...
+ Edge (4 6)[ 2] inserted.
+ Edge (0 1)[ 3] inserted.
+ Edge (4 5)[ 4] inserted.
+ Edge (1 3)[ 5] inserted.
- Edge (0 3)[ 6] not inserted.
+ Edge (2 5)[ 8] inserted.
+ Edge (3 4)[ 9] inserted.
Heap[1~4]: [10] [12] [17] [14]
```

```
*** MST: Prim Algorithm ***
[0] to [1] :   3
[1] to [3] :   5
[3] to [4] :   9
[4] to [6] :   2
[4] to [5] :   4
[5] to [2] :   8

*** MST: Sollin Algorithm ***
# of Trees: 7:
        (0  1)[3]
# of Trees: 6:
        (2  5)[8]
# of Trees: 5:
        (1  3)[5]
# of Trees: 4:
        (4  6)[2]
# of Trees: 3:
        (4  5)[4]
# of Trees: 2:
        (3  4)[9]
# of Trees: 1:
```